**Binary Exploitation:**

stonks:

Well, time to try something new. Binary exploitation. First started out with getting into the port.

Then I looked into the YouTube videos that were provided with the document. After sieving through those documents, I went ahead and looked at the C code that was given in the challenge.

Okay. So I was taught that in Binary Exploitation, you just abuse the code of the program. This is absolutely lovely, just my type of thing. Time to break the code as much as possible.

In this one, I saw that I didn’t know what was the first step to take. So I went back to the guru, the internet and tried deciphering what each code does in the different things.

Since there was a buffer limit and our task was to break that, I started spamming %x to break the buffer and what the code intended on doing.  
On doing so, I got a random hex. This random hex I put into a convertor to get a random sequence of characters.  
  
Now I didn’t know what to do with this random string of characters, so it was time to ask the help of YouTube. In the tutorial, the guy stated that he saw that the random string of characters, in it, for every four characters was forming a proper string… therefore, he wrote a python code (which I promptly just tried doing on my own but ended up just copying) and viola, I got the end result, which got me the flag.

References:

<https://www.youtube.com/playlist?list=PLhixgUqwRTjxglIswKp9mpkfPNfHkzyeN>

<https://www.youtube.com/watch?v=5-ZQubBWz3c&list=PLHUKi1UlEgOIc07Rfk2Jgb5fZbxDPec94&index=3>

<https://www.youtube.com/playlist?list=PL-DxAN1jsRa_rO0pr-1uJR_k_5Y9n-V1i>

<https://www.youtube.com/watch?v=ctpQdH-GGqY>

<https://www.geeksforgeeks.org/use-fflushstdin-c/>

<https://www.rapidtables.com/convert/number/hex-to-ascii.html>

Answer: picoCTF{I\_l05t\_4ll\_my\_m0n3y\_a24c14a6}

babygame01:

I gave the next one a try. I started the instance and it gave me the port address. Once I got in I saw this entire pattern.



At first my thought was to get to the X mark. After getting to the X mark, it just stated that I won and now I don’t know what to do.



There’s no hex or anything, so I really don’t know where I went wrong. Plus this time, there’s no source code to work with, so it’s back to reading writeups and seeing YouTube videos on how to do this.

After reading through some of them, it stated that I needed to have reverse engineered the code of this particular program, so I tried finding a site that would reverse engineer the program. After failing to find it, I ended up resorting to viewing some of the writeups and seeing their code itself.

After seeing that the code had nothing wrong, I resorted back to YouTube and seeing what was wrong and what was the approach that was meant to be taken.



I am absolutely not enjoying this type. But let’s see what can be done with this.

When I was just messing around, I noticed that I went out of bounds, weird things started to happen. 

At this stage, I just tried seeing what ASCII character was 64, and it was the ‘@’ symbol, which was the symbol used to depict that we’ve won, because we are at that position. Therefore I tried putting in p (because it was told in the video, I am trying to find out why p was input) and it solved the entire thing by itself. Now I need to figure out the significance of using p as the character.



Okay this turns out that this would give us the win condition directly, instead of trying to figure out how to solve it by ourselves.

It was working with how memory locations and everything worked. In the end, I followed the YouTube video entirely and tried solving. I might have not done well in this one, but at least I tried learning something.

References:

<https://www.youtube.com/watch?v=I9BL3fZOj1M>

<http://sticksandstones.kstrom.com/appen.html>

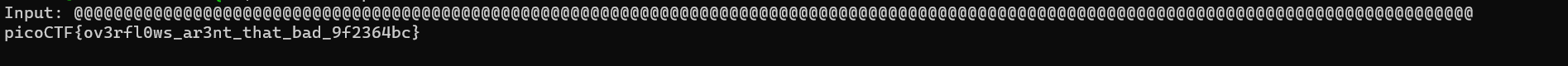
<https://www.google.com/search?q=reverse+engineering+from+port+address&rlz=1C1VDKB_enIN1065IN1065&oq=reverse+engineering+from+port+address&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQIRigATIHCAIQIRigAdIBCTE0Mjg1ajBqN6gCALACAA&sourceid=chrome&ie=UTF-8>

Answer: picoCTF{gamer\_m0d3\_enabled\_f4f6ad7d}

buffer overflow 0:

In this, I downloaded the C program and started debugging it on my own. In the first few lines, I noticed that it was stopping at 64 characters. Therefore, I tried seeing what would happen if I broke that.

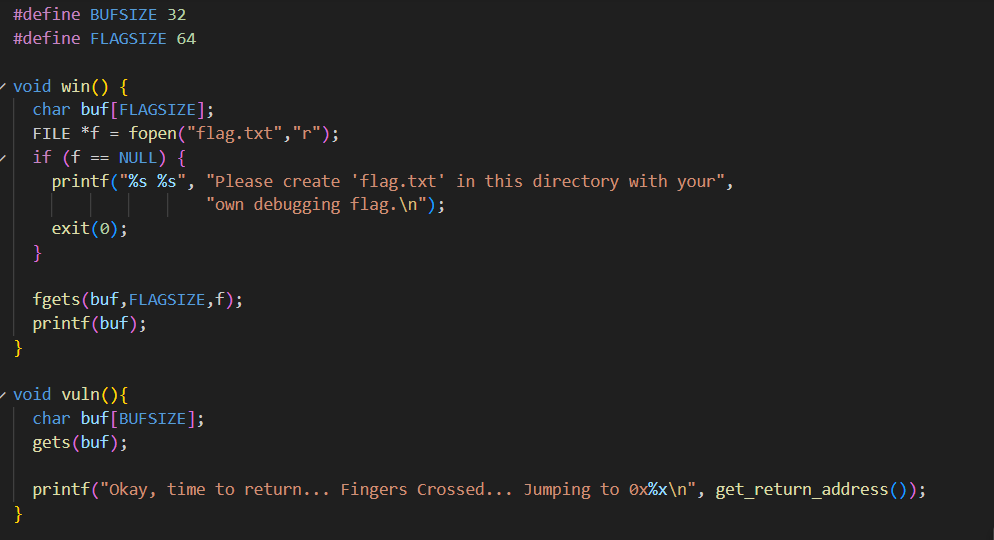
For some reason, it just output the answer, which I am just genuinely surprised about. Surely, it cannot be this simple?



And sure enough, it was the right answer. I still have no clue if it was that easy to break this program, but I am glad that this worked, because I ended up getting the answer.

Answer: picoCTF{ov3rfl0ws\_ar3nt\_that\_bad\_9f2364bc}

buffer overflow 1:

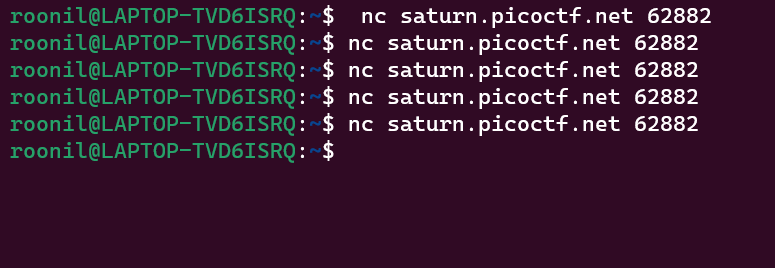


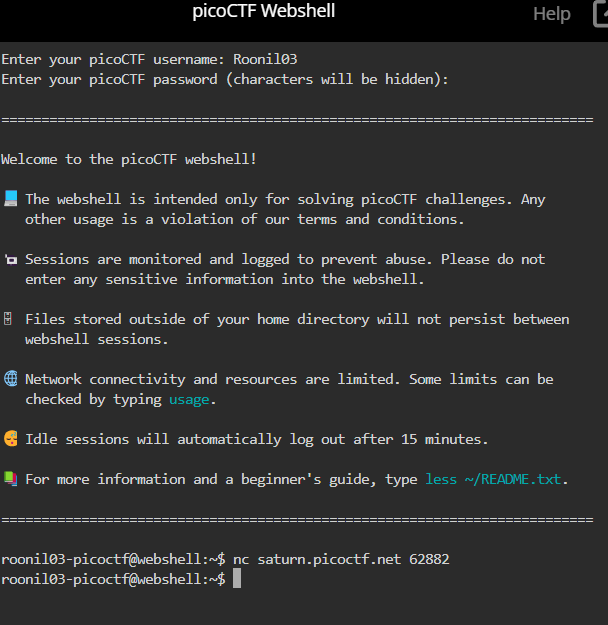
This time they have defined the BUFSIZE and FLAGSIZE differently, let’s see how we can approach this.

The hint states that I should read up on big and little Endian, so it was time to look at it. It also stated that different pointer addresses should call in different functions.

After reading up on this given information, I sought out another youtube video to see if it can make things simpler for me. After watching the video, I thought of what to do, and then continued with some of the instructions that were provided in the video. I still don’t know why I am struggling this much, but let’s see what can be done.

Now I have encountered this issue where I am just unable to get into the document.



It is just not letting me run the program.   
I am using the picoCTF version of the webshell and it is still not working. I have no idea where it is going wrong.

Okay it turns out that the instance time limit was over and it had shut down.

I tried understanding how to write the code and everything, but I am slowly realizing that I am not at all getting the gist of Binary Exploitation. I should visit this one once I have watched some videos and installed some more packages which a lot of people have been using in their projects.